

## **Ken Sturrock's SkyX Python Library Functions**

August 03, 2018

**Note:** Most parameters passed to these functions should be strings and enclosed in quotes. If in doubt: Use the source, Luke.

### **adjAGExposure(origAGExp, origAGDelay, XCoord, YCoord)**

This routine analyzes a guide star to determine if it is too bright or too dim for the supplied exposure duration. If the guide star is between 20% and 80% of the camera's range, then it does not suggest another value. Otherwise, it will scale the exposure to bring the guide star within this range. If it shortens the exposure, it will pad the delay with the difference. If it lengthens the exposure, it will not increase the exposure by more than 50%. The procedure will not try to measure a star unless the guider uses autodark or a calibration library.

Regardless of if a new exposure is created or the original is passed, the function will return an exposure and delay separated by a comma.

### **atFocus2(target, filterNum)**

Calls @Focus2 (requires "full auto mode") through the specified filter and then Closed Loop Slews back to the specified target. If the target is near the meridian, the routine will slew the mount east or west, away from the meridian, so that @Focus2 won't choose a star on the other side of the meridian, which would require a flip. Results are printed as a note and "Success" or "Fail" is returned.

### **atFocus2Both(host, target, filternum)**

Calls @Focus2 on both a local and remote Imaging camera. It's necessary to have this special version because @Focus2 depends on slewing to a focusing star. This routine slews to the star, focuses and then asks the remote focuser to focus before slewing back to target. It ASSumes that you will want to focus both cameras through the same filter number.

You must also take some care to calibrate the second (remote) rig on the same magnitude of focusing star as the primary (local). That way the remote rig can use the same star as selected by the local rig. You'll need to keep the remote machine's "virtual mount" synched with the local mount.

### **atFocus3(target, filterNum)**

Calls @Focus3 through the specified filter. If a target is specified then it will Closed Loop Slew back to the target about 50% of the time (randomly). The reason why it might be a good idea to slew back to target is to serve as an occasional reset for any dithering pattern and to fix any unexpected centering issues. If you don't want to re-slew back to the target then specify the target as "NoRTZ". Results are printed as a note and "Success" or "Fail" is returned.

### **atFocusRemote(host, whichCam, method filter)**

A generalized routine to focus either camera (Imager or Guider) with either @Focus2 or @Focus3. It does not return to target if using @F2 because it assumes that you'll handle that with the main script.

### **calcImageScale(whichCam)**

This function calculates the image scale for the specified camera. If no images are available, it will take an image in order to extract the pixel size value from the FITS header. If there are problems extracting the needed information from the FITS header, it will let you know and default to 1.7 AS/Pix (The DSS standard). The guider settling limit calculator and dither routine use this function. The function will return the image scale in AS/Pixel

### **calcSettleLimit()**

Compares the imaging camera and guiding camera image scales and suggests a percent of the guider pixel that represents an imaging pixel. It has some modifications for my Takahashi Temma mounts or for highly under-sampled guiders. The function will return the settle threshold in guider camera pixels.

### **camConnect(whichCam)**

Connects the specified camera ("Imager" or "Guider"). "Success" or "Fail" is returned.

The remote version does this task on a remote system.

### **camDisconnect(whichCam)**

Disconnects the specified camera. "Success" or "Fail" is returned.

The remote version does this task on a remote system.

### **cloudWait()**

Turns off the sidereal drive and waits five minutes. It will then “wake up” every five minutes to see if it can detect stars with the guider. If no stars are to be found after half an hour, it will hardPark().

### **CLSlew(target, filterNum)**

Closed Loop Slews to the specified target using the specified filter. It performs the CLS in two stages: an initial slew, followed by the actual CLS. This was done to cope with very slow moving mounts. Results are printed as a note. “Success or “Fail” is returned.

### **dither()**

Dithers the mount using the “jog” technique (versus the “move guide star” technique). The dither directions and distance are random but adjusted based on image scale and declination. The smallest it will dither is about a pixel distance. The distances and directions are printed.

### **findAGStar()**

This module is a re-packaged Javascript routine that analyzes the current guider image and searches for a guide star based on size, shape, location and brightness. Colin McGill & Kym Haines both provided critical assistance and ideas for the routine. It returns X & Y coordinates and a path to the analyzed file.

If you really want to understand what it does, please read through the diagnostic version in my original Bash-based script set.

### **getActiveImagePath()**

This routine returns the path to the currently active image.

### **getStats()**

This provides some basic pointing, FWHM & focuser position statistics. It uses Image Link, so it is a bit slow. It analyzes the current imaging camera image. “Success or “Fail” is returned.

The remote version does this task on a remote system.

### **getStatsPath(imgPath)**

This routine runs statistics on a specified file (full path) as specified by imgPath. It's a bit of a specialized routine which is useful for analyzing groups of previously shot (or modified) images.

### **getTemp()**

Provides the current selected temperature sensor temperature.

### **hardPark()**

Slews the mount towards the appropriate pole, turns off the sidereal motor, resets the camera defaults and disconnects the cameras.

### **isDayLight()**

Tells you if it is light outside and, if it is evening twilight, will wait for darkness.

### **isGuiderLost(limit)**

Determines if the autoguider is lost. It simply takes the supplied guider pixel error limit and multiplies it by three. Returns a "Yes" or a "No".

### **preRun()**

Uses the "mysterious" INI file variables to make sure that the user has needed configuration values set. "Success or "Fail" is returned.

### **remoteImageDone(host, whichCam)**

Determines if the image executed on a remote system has completed. This is needed in case you execute the remote image asynch.

### **reSynch()**

Synchronizes the mount position and the sky chart.

### **softPark()**

Turns off the sidereal drive (not on the simulator) and exits the script. It can be used for pausing the mount for safety if you don't want to do a hardPark.

### **settleGuider(limit)**

Measures the guider error until it reports less than the supplied limit five times in a row. If the error remains higher than the limit for longer than 30 checks, it will go ahead and continue anyway. If the guider appears lost, it will return the status “Lost” so that your main script can do something intelligent.

Just remember, your guider will guide as good as it can. The only purpose of the limit & settle functions is to allow the process to continue sooner rather than waiting some arbitrary time. Returns “Settled” or “Lost”.

### **slew(target)**

Slews to the specified target. It uses the same rules as the “find box” and you can specify names without spaces (e.g. ngc363 ).

The remote version slews a remote (real or virtual) mount.

### **startGuiding(exposure, delay, XCoord, YCoord)**

This function starts guiding with the specified exposure and delay on the star specified by the two coordinates.

### **stopGuiding()**

Stops guiding.

### **takeImage(whichCam, exposure, delay, filterNum)**

Takes an image with the specified camera (“Imager” or “Guider”) for the specified exposure, with the specified delay through the specified filter. Use “NA” if there are no filters or you don’t care. Remember that the first filter in the wheel is filter ZERO and not one. “Success or “Fail” is returned.

The remote version runs this task on a remote host.

### **targAlt(target)**

Returns the specified target’s altitude.

### **targExists(target)**

Makes sure that the target exists in the SkyX database. Returns “Yes” or “No”.

**targHA(target)**

Returns the target's Hour Angle. This is used primarily for determining if the mount needs to flip.

**timeStamp(message)**

This is a quick way to print an informational message with a time stamp.

**TSXSend(message)**

This is the core function. It sends the specified message to SkyX across the network socket and returns the result. Note the "Verbose" variable at the top of the file.

The remote version runs the command on a remote machine with a different IP address & port.